



# Mimicking Robotic Arm

Written By: Andrew Kaye

## TOOLS:

- [Acrylic cement \(1\)](#)
- [Digital Calipers \(1\)](#)
- [Hot Glue gun & hot glue \(1\)](#)
- [Laser cutter or jigsaw, router, or coping saw \(1\)](#)
- [Screwdrivers \(1\)](#)
- [Soldering iron \(1\)](#)

## PARTS:


- [Arduino Uno \(1\)](#)
- [Breadboard kit \(1\)](#)
- [9 volt battery and clip \(1\)](#)
- [Batteries, AA \(4\)](#)
- [Battery holder \(1\)](#)  
*For 4 AA batteries*
- [Potentiometer \(4\)](#)
- [Servo \(generic\) \(4\)](#)  
*2 large and 2 small*
- [Jumper wires \(1\)](#)
- [Wire \(1\)](#)
- [12"x12"x1/8" Acrylic \(2\)](#)
- [Screws \(1\)](#)  
*Various sizes and types*
- [Very thin plastic sheet \(1\)](#)

## SUMMARY

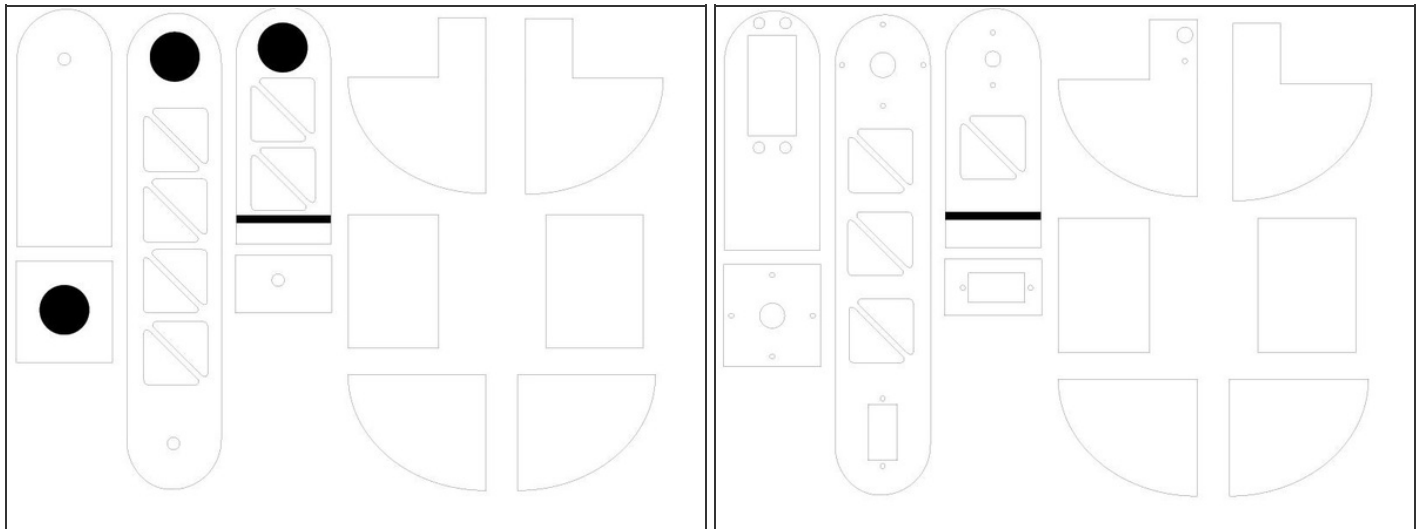
In this project you will build two robotic arms. One has potentiometers on the joints, and the other has servos. When you manipulate the potentiometer arm, the servo arm mimics it. An Arduino provides the control to make it all happen.



## Step 1 — Gathering parts and potentiometers



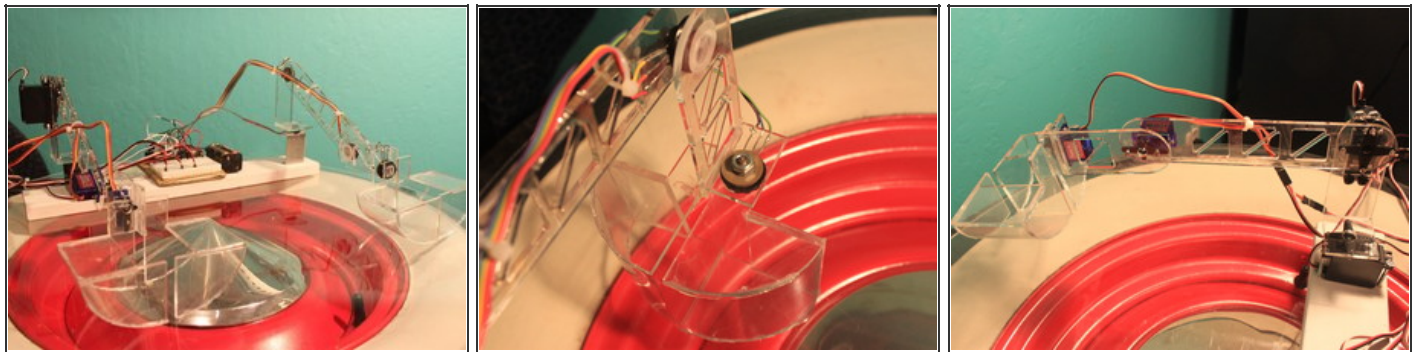
- Find four matching potentiometers. I recommend ones that have a large flat surface so you can easily glue them to the arm pieces.
- The Arduino should read most potentiometers as values of 0 to 1023. If you find your arms to be not very responsive to your movements, check the values read from the potentiometers using the serial monitor. I found that the Arduino read the potentiometers I used as values from 21 to 589 so I had to map my values differently. 
- Wire your potentiometers to the breadboard. I had all the positive and negative leads going to one set of power rails. The signal wires should go to the analog inputs. Write down or remember which is which if you want to save time when doing the code.
- If you find the servo moves the wrong way, try reversing the polarity of the potentiometer




## Step 2 — Designing and cutting the arms



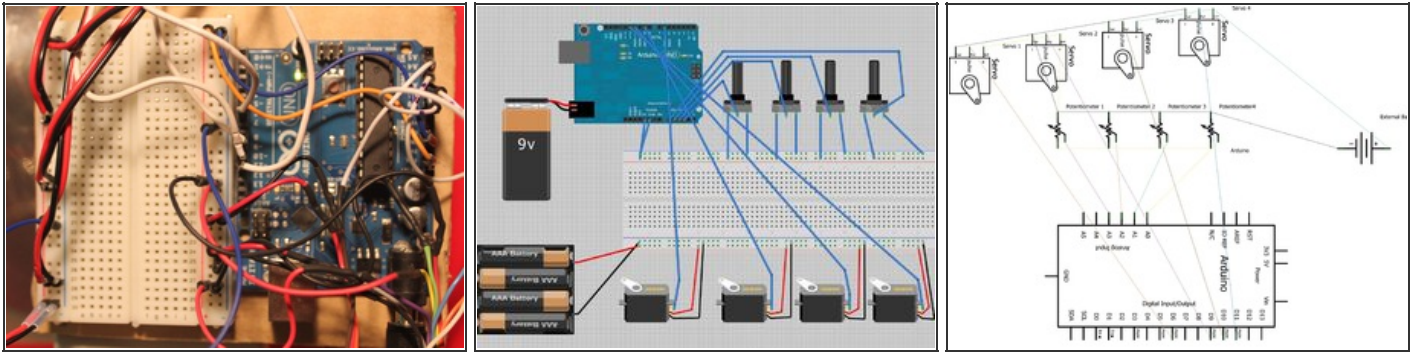
- Using calipers, measure out where the mounting holes should be for your servos. I also rastered in circles for potentiometer placement and a line for gluing on the bracket for the claw.
- The bracket for the claw is precisely positioned so that the edges of the claw line up due to the thickness of the servo mount or the potentiometer mount. 
- I also put in cutouts to reduce weight. Be aware of the weight of the arm; I found that the servo that lifts the large arm segment barely had enough torque to lift the arm. 
- After the arms are designed, cut them using whatever methods are available to you. I have access to a laser cutter so that is what I used.

### Step 3 — Assembly



- Now that you have all your pieces, it is time to assemble the arms. Keep track of which pieces are for which arm and assemble the arms as in the pictures. I recommend doing any gluing first and then mounting servos and potentiometers.
- I used a combination of hot glue and acrylic cement to assemble my arms. For lower joints that need a lot of strength I would use the acrylic cement. 
- When mounting the potentiometers keep in mind their orientation. They need to be "facing" the same side or the servos will move opposite to your movements of the potentiometer arms. 
- After I had the claw halves glued I cut out a piece of the very thin plastic to make the bottom. I found it easier to glue one end and cut to fit rather than measuring the curve for a dimension. 

## Step 4 — Wiring



- The wiring for this is not that complicated. Most of it is just distributing power and running signal wires.
- The positive and negative wires from the servos should go to the power rails along with the wires from the external battery pack.
- The positive and negative wires from the potentiometers should go to the other set of power rails on the breadboard.
- The +5 and Ground pins from the Arduino should go to the power rails of the breadboard where the potentiometers are connected.
- The signal wires from the servos should go to their corresponding digital output pins on the Arduino (these need to be PWM outputs).
- The signals from the potentiometers should go to their corresponding analog inputs.
- I jumpered the ground rails together to eliminate noise. I'm not sure if this is necessary for it to work, but it doesn't hurt.



## Step 5 — The code

```

arm | Arduino 1.0.1
File Edit Serial Tools Help
arm
servo lower; //creates object for the lower servo of the arm
int potpin0 = 0; //gives the analog pin for the corresponding potentiometer
int val0; //makes a variable for the value of the potentiometer

servo claw;
int potpin1 = 1;
int val1;

servo clawpin2;
int potpin2 = 2;
int val2;

servo base;
int potpin3 = 3;
int val3;

void setup()
{
  lower.attach(10); //tells the arduino which digital pin the servo is connected to
  claw.attach(11);
  clawpin.attach(9);
  base.attach(6);
}

void loop()
{
  val0 = analogRead(potpin0); //sets the variable equal to what the current value of the potentiometer is
  val0 = map(val0, 21, 509, 0, 179); //scales this value to the range of the servo
  lower.write(val0); //writes the value to the lower servo
  delay(15); //gives the servo time to get to the position

  val1 = analogRead(potpin1);
  val1 = map(val1, 21, 509, 0, 179);
  claw.write(val1);
  delay(15);

  val2 = analogRead(potpin2);
  val2 = map(val2, 21, 509, 0, 179);
  clawpin.write(val2);
  delay(15);

  val3 = analogRead(potpin3);
  val3 = map(val3, 21, 509, 0, 179);
  base.write(val3);
  delay(15);
}

```

- The code that I used to control the arms is very similar to the "servo knob" example that comes with all installations of the Arduino IDE.
- Variables are created for each servo and potentiometer.
- The code reads the value of the potentiometer, maps it to the range of the servo and writes it to the servo in a loop.
- See the attachments to this project for a PDF file of the code. You can copy and paste it into the Arduino IDE.

Other parts about this project that I would've liked to include but didn't have time to are adding code to make the servo arm move smoothly without twitching and using servos that have the ability to move a full rotation.

This document was last generated on 2013-02-12 12:41:40 PM.